Object Oriented Programming

Everything Is an Object

The progress of abstraction

1. Machine language
2. Assembly language
3. Procedural-Oriented languages (BASIC, C, etc…): Top-down
4. Object-Oriented language: Bottom-up

OOP characteristics

- Everything is an object.
- A program is a bunch of objects telling each other what to do by sending messages.
- Each object has its own memory made up of other object.
- Every object has a type.
- All objects of particular type can receive the same messages.
- An object has state, behavior and identity.

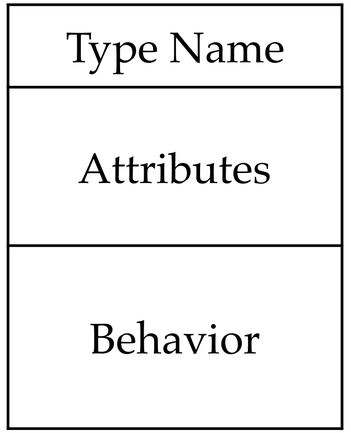
Classes

A set of objects that have identical characteristics and behaviors.

- A description of the common properties of a set of objects.
- A concept.
- A class is part of the program.

Pseudo Code

Unified Modeling Language (UML)



Object

A self-contained entity that consists of both data and procedures to manipulate the data.

- A representation of the properties of a single instance.
- An object is part of data and program execution.

An object provides services

- High Cohesion
- Low Coupling

Encapsulation

Class Creators & Client Programmers

Access control

- Classes
- Packages
- Nested objects and classes

Access specifier

- public
- protected
- default
- private

Inheritance

A feature that represents the “IS-A” or “IS-LIKE-A” relationship between different classes.

Generalization and Specialization

Ex. Language & English

Polymorphism

One name, many forms & Dynamic binding: How polymorphism is implemented. If an instance sends a stimulus to another instance, but does not have to be aware of which class the receiving instance belongs to, we say that we have polymorphism.

- Overriding: Also called run-time polymorphism. Method will be used for method overriding is determined at runtime based on the dynamic type of an object.

- Overloading: Which is referred to as compile-time polymorphism. For method overloading, the compiler determines which method will be executed, and this decision is made when the code gets compiled.

Parameterized types

Generic programming is a style of computer programming in which algorithms are written in terms of types to-be-specified-later that are then instantiated when needed for specific types provided as parameters.

Wildcards vs. Generic vs. Object

Reusing the implementation

Type of Relationship (Weak to Strong)

- Association (): Each of these objects have their own life-cycle and there is no owner.
- Aggregation (): ”HAS-A”. Specialized form of association.
- Composition (): “PART-OF”. Special form of aggregation.
- Realization (): Relationship between interface and implementing class.
- Generalization/Inheritance (): “IS-A”. Base-class and Derived-class.

Thank you